Árvores AVL
(Adelson-Velskii and Landis)

Universidade Federal do Amazonas
Departamento de Eletrônica e Computação

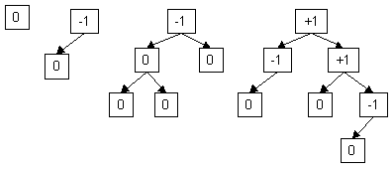
Introdução (1)

- Árvore Balanceada
  - Uma árvore binária balanceada é aquela em que, para qualquer nó, suas sub-árvores esquerda e direita têm a mesma altura
- Árvore AVL
  - Uma árvore binária de busca é balanceada quando, para cada nó, as alturas de suas subárvores (sa) esquerda e direita **diferem de, no máximo, 1**
  - Essa diferença é chamada *fator de balanceamento,* ou `FB(n)`

Introdução (2)

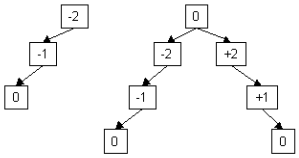
- Seja um nó **n** qualquer da árvore:
  - `FB(n)` = `altura`(sad) – `altura`(sae)
  - se `FB(n)` = 0, as duas sub-árvores têm a mesma altura
  - se `FB(n)` = -1, a sub-árvore esquerda é mais alta que a direita em 1
  - se `FB(n)` = +1, a sub-árvore direita é mais alta que a esquerda em 1

Introdução (3)



Exemplos de Árvores AVL

Introdução (4)



Exemplos de Árvores Não-AVL

Introdução (5)

- A vantagem da árvore AVL sobre uma degenerada está na eficiência das suas operações de busca
  - Sendo a altura da AVL bem menor, o número necessário de comparações diminui sensivelmente
  - Numa árvore degenerada de 10.000 nós, são necessárias, em média, 5.000 comparações, numa busca; numa árvore AVL, com o mesmo número de nós, essa média baixa para 14
- O algoritmo deve, a cada inserção, fazer as correções necessárias para garantir que qualquer nó **n** tenha `|FB(n)|` <= 1

Balanceamento (1)

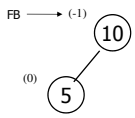
- Como fazemos então para manter uma árvore AVL balanceada?
  - Inicialmente inserimos um novo nodo na árvore normalmente
  - A inserção deste novo nodo pode ou não violar a propriedade de balanceamento
  - Caso a inserção do novo nodo não viole a propriedade de balanceamento podemos então continuar inserindo novos nodos
  - Caso contrário precisamos nos preocupar em restaurar o balanço da árvore
  - A restauração deste balanço é efetuada através do que denominamos de rotações na árvore

Balanceamento (2)

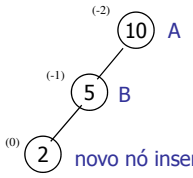
- Serão usados dois ponteiros **A** e **B**, para auxiliar:
  - **A** é nó ancestral **mais próximo** do nó inserido com `FB(nó)` ≠ 0 antes da inserção
    - ou a própria raiz se não há nenhum nó com `FB(nó)` ≠ 0 (antes da inserção) no *caminho da busca*
  - **A** é também chamado de **Pivô**
  - **B** é filho de **A** na sub-árvore onde ocorreu a inserção
  - Considerar **ligações unidirecionais** entre pai e filho

Exemplo de Desbalanceamento

Antes da Inserção do valor 2



Após a Inserção do valor 2



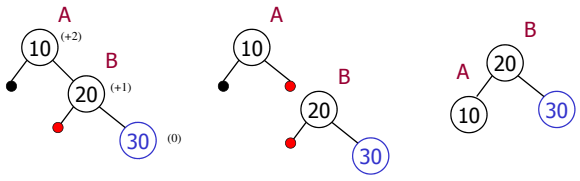
Quem é A e quem é B?

Rotação Simples

```
proc rotação simples
   se FB(A) = +1          // antes da inserção
      então rotação simples à esquerda
   senão rotação simples à direita
   fim se
   zera fatores de A e de B
fim proc
```

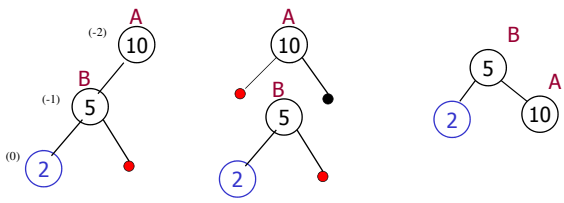
Rotação Simples à Esquerda

```
A->dir = B->esq;
B->esq = A;
```



Rotação Simples à Direita

```
A->esq = B->dir;
B->dir = A;
```

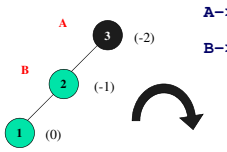


Exercício: Inserção (1)

- Mostrar as rotações necessárias para a construção da seguinte árvore AVL: 3, 2, 1, 4, 5, 6 e 7

Exercício: Inserção (2)

- Mostrar as rotações necessárias para a construção da seguinte árvore AVL: 3, 2, 1, 4, 5, 6 e 7

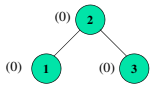



```
A->esq = B->dir;
B->dir = A;
```

Quem é A? Quem é B? Quais os FB's?
O que é necessário fazer para equilibrar essa árvore?

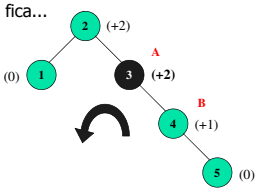
Exercício: Inserção (3)

O resultado da rotação à direita fica...




```
B->pai = A->pai;
A->dir = B->esq;
B->esq = A;
```

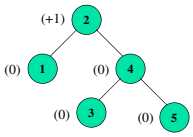
Após a inserção de 4 e 5 fica...



O que tem que ser feito para reequilibrar?

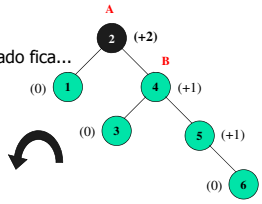
Exercício: Inserção (4)

O resultado da rotação à esquerda fica...




```
B->pai = A->pai;
A->dir = B->esq;
B->esq = A;
```

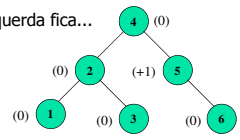
Mas quando o **6** é inserido o resultado fica...



O que tem que ser feito para reequilibrar?

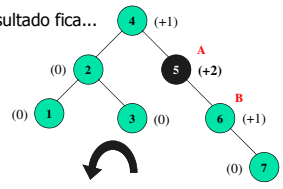
Exercício: Inserção (5)

O resultado da rotação à esquerda fica...




```
B->pai = A->pai;
A->dir = B->esq;
B->esq = A;
```

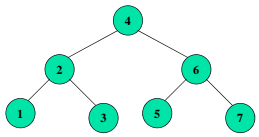
Mas quando o **7** é inserido o resultado fica...



O que tem que ser feito para re-equilibrar?

Exercício: Inserção (6)

O resultado da rotação à esquerda fica...



Rotação Dupla

proc rotação dupla
  se FB(A) = +1 // antes da inserção
    então rotação dupla à direita
    senão rotação dupla à esquerda
  fim se
  ajusta fatores dos nós envolvidos na rotação
fim proc

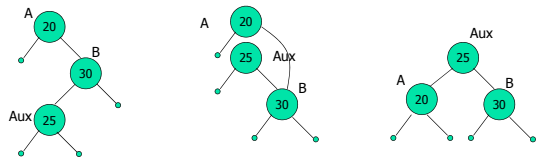
Rotação Dupla à Direita (1)

- É composta por uma rotação simples à direita (B e Aux) seguida de uma rotação simples à esquerda (A e Aux)
- Aux é o filho esquerdo de B

Rotação Dupla à Direita (2)

```
Aux = B->esq;
// rotação simples à direita (B - Aux)
B->esq = Aux->dir;
Aux->dir = B;

// rotação simples à esquerda (A - Aux)
A->dir = Aux->esq;
Aux->esq = A;
```



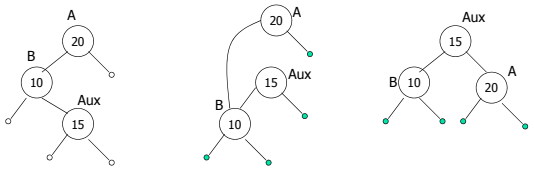
Rotação Dupla à Esquerda (1)

- É composta por uma rotação simples à esquerda (B e Aux) seguida de uma rotação simples à direita (A e Aux)
- Aux é o filho direito de B

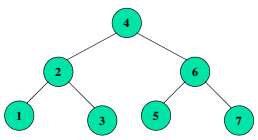
Rotação Dupla à Esquerda (2)

```
Aux = B->dir;
// rotação simples à esquerda (B - Aux)
B->dir = Aux->esq;
Aux->esq = B;

// rotação simples à direita (A - Aux)
A->esq = Aux->dir;
Aux->dir = A;
```

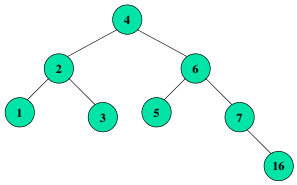


Exemplo: Rotação Dupla (1)



Como ficaria se fosse inserido o valor 16?

Exemplo: Rotação Dupla (2)



A Árvore ainda fica OK!

Como ficaria se fosse inserido o valor 15?